Implementasi Steganografi pada File GIF dengan *Extended Gifshuffle*

Faris Hasim Syauqi - 13519050
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519050@std.stei.itb.ac.id

***Abstract*—File GIF merupakan sebuah format file terdiri dari beberapa frame yang dapat ditampilkan secara berurutan. Salah satu algoritma steganografi pada file GIF adalah Gifshuffle. Gifshuffle bekerja dengan cara melakukan permutasi pada tabel warna, dimana setiap urutan warna yang dihasilkan dapat merepresentasikan bilangan tertentu. Pada algoritma ini, terdapat persoalan *capacity*, yaitu banyaknya pesan yang dapat disembunyikan tidak lebih dari 210 byte. Pada penelitian ini, akan diajukan algoritma *Extended Gifshuffle* yang merupakan ekstensi dari algoritma Gifshuffle. *Extended Gifshuffle* diajukan dalam rangka memperbesar kapasitas pesan yang dapat disembunyikan dengan cara memanfaatkan *Local Color Table* yang ada pada setiap frame dan melakukan duplikasi frame.**

Keywords—GIF; steganografi; Gifshuffle; capacity; Extended Gifshuffle;

I. Pendahuluan

Steganografi adalah ilmu atau seni untuk menyembunyikan pesan rahasia dengan suatu cara sedemikian sehingga tidak seorang pun yang mengetahui keberadaan pesan tersebut[1]. Tujuan dari steganografi adalah untuk menyembunyikan keberadaan pesan yang akan dikirimkan sehingga pihak lawan tidak mengetahui adanya pesan tersebut. Steganografi digital termasuk ke dalam steganografi modern dengan cara menyembunyikan pesan digital di dalam dokumen digital lainnya. Makalah ini akan berfokus pada penyembunyian pesan digital ke dalam file berformat GIF.

Ada beberapa algoritma yang sudah dikembangkan untuk melakukan steganografi pada file GIF, diantaranya yaitu EzStego dan Gifshuffle. Algoritma EzStego adalah algoritma steganografi pada file GIF dengan metode *Least Significant Bit (LSB)* disertai dengan pengurutan pada tabel warna sehingga warna yang berdekatan ada pada posisi yang dekat, sehingga perubahan LSB tidak memberi dampak signifikan. Adapun Gifshuffle bekerja dengan cara melakukan permutasi pada tabel warna, dimana setiap urutan warna yang dihasilkan dapat merepresentasikan bilangan tertentu. Pada makalah ini akan berfokus pada pembahasan algoritma gifshuffle.

Algoritma Gifshuffle yang sudah ada saat ini memiliki kapasitas penyembunyian pesan yang terbatas yaitu hanya sebesar 210 byte. hal ini dikarenakan warna unik maksimal yang dapat disimpan pada tabel warna hanyalah 256 warna. Salah satu kriteria steganografi yang baik adalah *capacity*. Pada makalah ini, akan diajukan ekstensi dari algoritma ini untuk memperbesar kapasitas dengan cara memanfaatkan *Local Color Table* yang ada pada setiap frame serta melakukan duplikasi frame.

II. Landasan Teori

A. *Steganografi*

Steganografi adalah ilmu atau seni untuk menyembunyikan pesan rahasia dengan suatu cara sedemikian sehingga tidak seorang pun yang mengetahui keberadaan pesan tersebut[1]. Tujuan dari steganografi adalah untuk menyembunyikan keberadaan pesan yang akan dikirimkan sehingga pihak lawan tidak mengetahui adanya pesan tersebut.

Perbedaannya dengan kriptografi, yaitu kriptografi menyembunyikan isi pesan sehingga pesan tidak dapat dibaca oleh pihak lawan, sementara steganografi menyembunyikan keberadaan pesan untuk menghindari kecurigaan dari pihak lawan.

Steganografi digunakan dalam berbagai aplikasi sehari-hari, misalnya, kontrol hak cipta materi, meningkatkan ketahanan mesin pencari gambar dan ID pintar (kartu identitas) dimana detail individu disematkan dalam foto. Penerapan lain dari steganografi adalah sinkronisasi video-audio, sirkulasi data rahasia perusahaan yang aman, siaran TV, paket TCP/IP (misalnya ID unik dapat disisipkan ke dalam gambar untuk menganalisis lalu lintas jaringan pengguna tertentu).

Steganografi digital termasuk ke dalam steganografi modern dengan cara menyembunyikan pesan digital di dalam dokumen digital lainnya[1]. Ada beberapa terminologi dalam steganografi digital:

- *Embedded message* atau *secret message*, merupakan pesan yang akan disembunyikan.
- *Cover-object*, media digital yang digunakan untuk menyembunyikan embedded message.
- *Stego-object* (*stego-data*), media yang sudah berisi pesan *embedded message*.
- *Stego-key*, kunci yang digunakan untuk menyisipkan pesan dan mengekstraksi pesan dari stego-object.

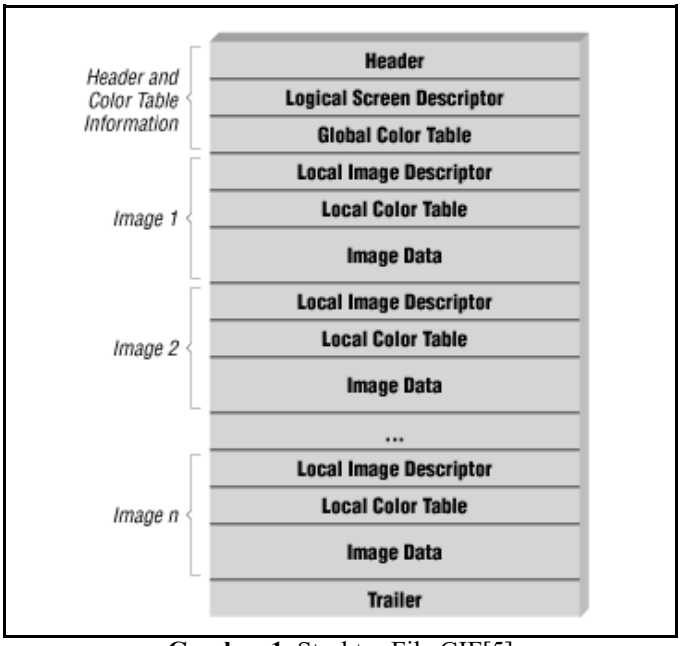
Steganografi yang bagus memiliki beberapa kriteria, yaitu sebagai berikut:

- *Imperceptible*, keberadaan pesan rahasia tidak dapat dipersepsi secara visual atau secara audial.
- *Fidelity*, kualitas *cover-object* tidak jauh berubah akibat penyisipan pesan rahasia.
- *Recovery*, pesan yang disembunyikan harus dapat diekstraksi kembali.
- *Capacity*, ukuran pesan yang disembunyikan dapat sebesar mungkin.

B. File GIF

Graphic Interchange Format yang biasa dikenal dengan GIF, merupakan sebuah format file gambar. Hal yang membedakan GIF dengan format gambar lainnya seperti JPEG dan PNG adalah GIF dapat terdiri dari banyak gambar yang disebut *frame* dan adanya mekanisme untuk *looping* setiap *frame,* sehingga setiap *frame* dapat ditampilkan secara berurutan dan berulang-ulang[2]. Selain itu, GIF juga dilengkap dengan *lossless compression* yaitu kompresi Lempel-Ziv-Welch (LZW)[3]. Gambar pada GIF termasuk ke dalam jenis *Indexed Image* yang artinya data gambar disimpan sebagai larik yang setiap elemennya merupakan indeks yang merujuk ke warna pada tabel warna[4].

Struktur file GIF terdiri dari urutan blok dimana dua blok pertama memiliki ukuran dan posisi yang tetap blok sisanya memiliki ukuran dan posisi yang bervariasi[3]. Dua blok pertama tersebut adalah *Header* dan *Logical Screen Descriptor.* Berikut ini merupakan struktur dari file GIF.



Gambar 1. Struktur File GIF[5]

Blok *Header* terdiri dari 3 byte *signature* dan 3 byte *version* dimana *signature* bernilai "GIF" dan *version* adalah "87a" atau "89a" tergantung versi GIF mana yang digunakan. Blok *Logical Screen Descriptor* merupakan blok yang memberikan informasi kepada *decoder* banyaknya ruang yang diperlukan. Blok ini terdiri dari ukuran canvas, *flag*, *background color index*, dan *pixel aspect ratio*. Blok selanjutnya adalah *Global Color Table* yang merupakan tabel warna yang dapat digunakan oleh setiap frame. setiap warna disimpan sebagai 24-byte RGB. Blok ini bersifat opsional karena setiap *frame* mungkin memiliki tabel warna lokal. *Global Color Table* dapat menampung maksimal 256 warna unik.

Blok *Global Color Table* kemudian diikuti dengan barisan frame dimana setiap frame nya terdiri dari 3 blok, yaitu *Logical Image Descriptor, Local Color Table,* dan *Image data. Logical Image Descriptor* berfungsi sebagai *descriptor* untuk frame tersebut, *Local Color Table* merupakan tabel warna yang digunakan oleh frame sedangkan *Image Data* merupakan data frame yang terkompresi dengan LZW. Sebelum dilakukan kompresi, pada awalnya data pada *Image Data* merupakan larik yang setiap elemennya merupakan indeks yang merujuk suatu warna di tabel warna. Jika *Local Color Table* tidak ada maka frame akan menggunakan *Global Color Table*. Sama seperti *Global Color Table*, blok ini juga mampu menampung hingga 256 warna unik.

C. Gifshuffle

Gifshuffle adalah program yang dikembangkan oleh Matthew Kwan yang dapat digunakan sebagai alat steganografi pada file GIF. Algoritma yang digunakan oleh gifshuffle adalah dengan cara melakukan *shuffling* pada tabel warna, sehingga gambar tidak akan berubah secara visual. gifshuffle dapat bekerja untuk semua file GIF, termasuk yang memiliki *transparency* dan *animation.* Gifshuffle juga menyediakan kompresi dan enkripsi untuk pesan yang akan disembunyikan[6].

konsep yang digunakan algoritma Gifshuffle dimulai dari persoalan permutasi. Untuk mengurutkan n buah barang yang berbeda maka akan ada n! buah cara. Konsep ini kemudian ditarik ke dalam ilmu steganografi dimana urutan n buah item dapat merepresentasikan bilangan dalam rentang [0..n!-1], dimana bilangan ini merepresentasikan pesan yang akan disembunyikan dengan panjang pesan maksimal log2(n!) bit. Misalnya, untuk tabel warna dengan banyaknya warna unik adalah 256, maka dapat menyembunyikan pesan dengan panjang 1683 bit (210 byte).

Berikut adalah langkah-langkah dalam algoritma Gifshuffle.

1. Dimulai dari pesan yang akan disisipkan. Pesan tersebut akan diubah ke dalam bentuk biner, menjadi barisan 1 dan 0.
2. Barisan tersebut kemudian akan merepresentasikan suatu bilangan, misalkan M.
3. Pada file GIF yang menjadi *cover-object,* hitung banyaknya warna unik pada tabel warna, misalkan N. kemudian hitung N-1!. Jika M > N-1!, maka pesan tidak bisa disisipkan, program berhenti.

4. Hilangkan warna duplikat, Urutkan setiap warna pada tabel warna berdasarkan rumus *red* * 65536 + *green* * 256 + *blue*.
5. Lakukan iterasi i dari 1..N, setiap warna ke-i di alokasikan ke posisi m mod i, kemudian m dibagi dengan i.
6. Setiap warna kemudian dimasukkan ke posisi yang telah dialokasikan pada tabel warna. Jika beberapa warna yang menempati posisi yang sama, maka warna yang sebelumnya akan digeser.
7. jika ukuran tabel warna lebih besar dibanding banyaknya warna unik, maka akan dilakukan padding dengan warna terakhir pada tabel warna yang awal.
8. *Image Data* kemudian dilakukan dekompresi dan setiap index pada *image data* akan disesuaikan dengan tabel warna yang baru, kemudian dilakukan kompresi ulang.

III. Analisis Permasalahan & Solusi

A. *Analisis Permasalahan*

Algoritma gifshuffle bekerja dengan cara melakukan *shuffling* pada tabel warna dimana setiap urutan warna yang dihasilkan dapat merepresentasikan bilangan dalam rentang [0..N-1!] dengan N adalah banyaknya warna unik. Banyaknya pesan yang dapat disembunyikan tergantung dari banyaknya warna unik pada tabel warna. Tabel warna pada GIF memiliki ukuran maksimum 256 warna, sehingga banyaknya ukuran pesan yang dapat disembunyikan tidak dapat lebih dari 1683 bit (210 byte). Jika suatu file GIF menggunakan kurang dari 256 warna unik, maka banyaknya pesan yang dapat disembunyikan menjadi lebih sedikit lagi.

Berikut adalah salah satu contoh dimana ukuran pesan yang akan disisipkan lebih dari 210 byte.

Gambar 2. Penyembunyian pesan melebihi kapasitas

Kemudian persoalan lainnya adalah besarnya ukuran *cover-object* tidak memberikan pengaruh apa-apa terhadap kapasitas pesan yang dapat disembunyikan. Jika ada dua file GIF yang masing-masing berukuran 1 MB dan 20 MB, Keduanya tetap hanya bisa menampung hingga 210 byte pesan rahasia.

Salah satu kriteria steganografi yang baik adalah *capacity*. Dengan demikian, dirasa perlu adanya ekstensi untuk algoritma ini sehingga dapat menampung pesan lebih dari 210 byte. Pada makalah ini, akan diajukan algoritma *Extended Gifshuffle* yang memungkinkan penyisipan pesan berukuran lebih dari 210 byte dengan memanfaatkan *Local Color Table* yang ada pada setiap frame juga melakukan duplikasi frame untuk memperbesar kapasitas pesan yang dapat disisipkan.

B. *Gambaran Umum Solusi*

Untuk mengatasi permasalahan yang telah dijelaskan sebelumnya, pada makalah ini akan diajukan algoritma *Extended Gifshuffle* yang mana merupakan algoritma Gifshuffle yang disertai dengan penggunaan *Local Color Table* serta penambahan frame duplikat untuk menambah kapasitas pesan yang akan disisipkan.

Secara umum, algoritma ini terdiri dari tiga fase, yaitu *Filling Phase*, *Growing Phase dan Shuffling Phase.* Setiap fase akan dijelaskan pada bagiannya masing-masing.

C. *Filling Phase*

Suatu file GIF mungkin saja hanya memiliki *Global Color Table* dan tidak ada satu pun frame yang memiliki *Local Color Table.* Kemudian *Global Color Table* tersebut juga mungkin saja hanya digunakan sebagian dari kapasitas maksimalnya, yaitu kurang dari 256 warna. Pada fase ini akan dilakukan penambahan *Local Color Table* sehingga setiap frame akan memiliki tabel warnanya sendiri. Kemudian pada fase ini juga akan dilakukan padding warna pada tabel warna sehingga tabel warna menggunakan kapasitas maksimalnya yaitu berisi 256 warna unik.

Berikut adalah langkah-langkah pada fase ini.

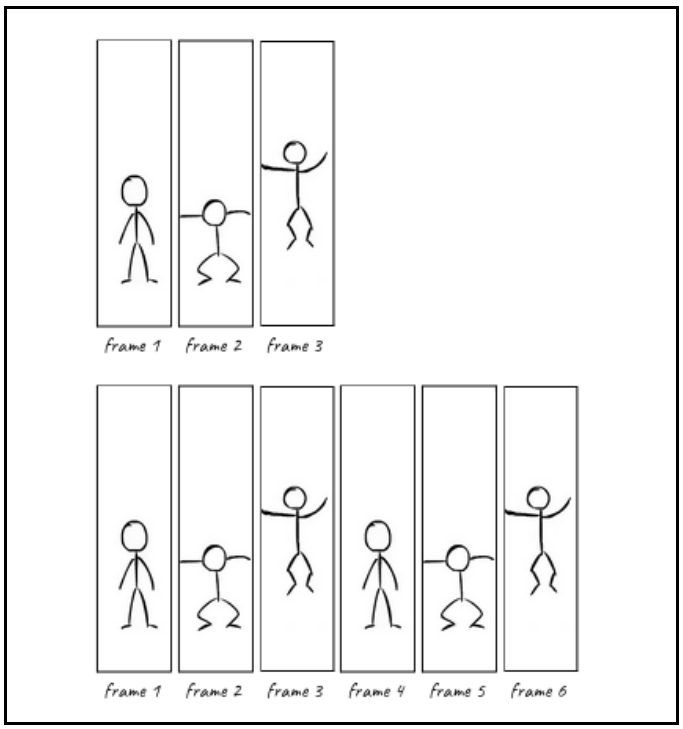
1. Periksa apakah GIF memiliki *Global Color Table.*
2. Jika tidak ada *Global Color Table*, maka salin *Local Color Table* pada frame pertama ke *Global Color Table.*
3. Hitung banyaknya warna unik pada *Global Color Table.* Jika kurang dari 256, maka tambahkan padding berupa warna acak yang belum ada pada tabel, hingga tabel berisi 256 warna unik.
4. Hitung banyaknya frame N.
5. Lakukan iterasi setiap frame F dari 1..N. Untuk setiap frame, periksa apakah frame tersebut memiliki *Local Color Table* atau tidak. Jika tidak memiliki *Local Color Table,* maka salin *Global Color Table* ke *Local Color Table* pada *frame* ini. Jika sudah ada *Local Color Table* pada *frame* ini, maka tambahkan padding warna acak jika tabel warna belum berisi 256 warna unik.

Pada akhir fase ini, untuk file GIF yang memiliki N buah frame maka akan memiliki N+1 buah tabel warna, dimana setiap tabelnya terdiri dari 256 warna unik.

D. *Growing Phase*

Setelah *Filling Phase*, file GIF yang menjadi *cover-object* mungkin saja masih belum memiliki kapasitas yang diperlukan untuk menyembunyikan pesan. Hal ini terjadi ketika panjang pesan M > (N+1) * 210 dimana N merupakan banyaknya *frame*. Untuk mengatasi hal ini, maka akan dilakukan duplikasi setiap frame, sehingga kapasitas dapat menjadi lebih besar. Dalam melakukan duplikasi, ada dua pendekatan yang dapat digunakan, yaitu *Vertical Grow* dan *Horizontal Grow*.

Untuk pendekatan *Vertical Grow*, semua frame akan diduplikasi, kemudian seluruh frame duplikat akan ditempatkan setelah seluruh frame asal. Penempatan ini dilakukan sebagai eksploitasi kondisi di zaman sekarang dimana *animated* GIF sering ditampilkan secara berulang-ulang. Ketika GIF ditampilkan, akan melakukan iterasi setiap frame, dan ketika iterasi mulai dilakukan terhadap *frame-frame* duplikat, akan ditampilkan seolah-olah diulang kembali dari frame pertama. Ilustrasi untuk pendekatan Vertical Grow dapat dilihat pada Gambar 3.

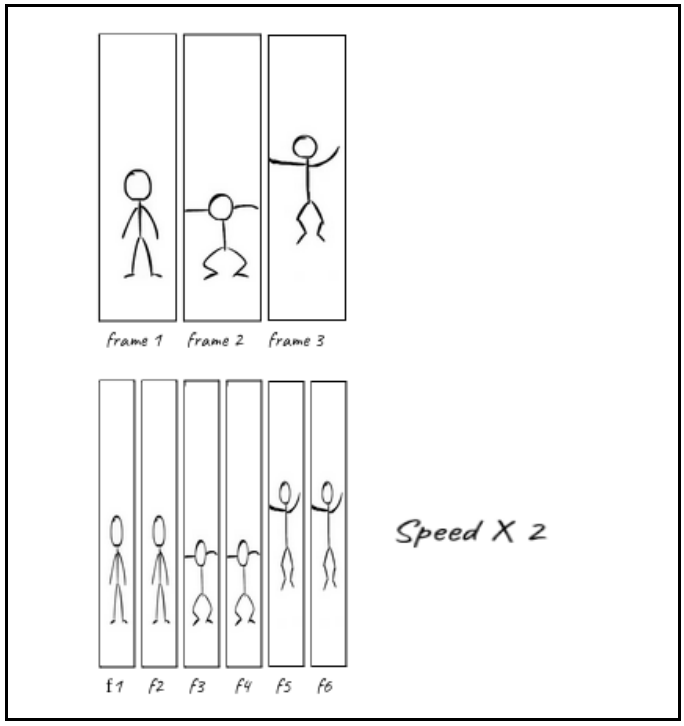


Gambar 3. Ilustrasi *Vertical Grow.* Atas : sebelum grow. Bawah : setelah grow

Kemudian pendekatan kedua adalah *Horizontal Grow,* dimana setiap frame akan diduplikasi dan setiap frame duplikat akan ditempatkan tepat setelah frame yang menjadi asalnya. Kemudian nilai delay akan diubah menjadi lebih singkat. Dengan demikian, ketika ditampilkan secara iteratif mulai dari frame pertama, meskipun setiap frame menjadi ditampilkan dua kali, hasilnya tidak akan berbeda dibanding sebelumnya karena waktu menampilkan setiap framenya menjadi lebih singkat. Namun untuk pendekatan ini, terdapat batasan mengenai nilai delay yang bisa dipilih pada file GIF, karena nilai yang dipilih harus merupakan *integer* dan tidak bisa dengan *float*. Ilustrasi untuk *Horizontal Grow* dapat dilihat pada Gambar 4.

Kemudian alternatif yang dapat dilakukan adalah dengan pendekatan *hybrid* dari dua pendekatan tersebut. Salah satu contohnya yaitu dengan melakukan *Horizontal Grow* selama delay masih bisa diubah sedemikian sehingga hasilnya tidak berubah secara visual. Kemudian ketika mengubah delay sudah tidak lagi memungkinkan, selanjutnya lakukan *Vertical Grow.*

Fase ini akan terus dilakukan selama *Cover-object* belum bisa menampung seluruh pesan yang akan disisipkan, yakni selama $M > (N+1) * 210$.



Gambar 4. Ilustrasi *Horizontal Grow.* Atas : sebelum grow. Bawah : setelah grow

E. Shuffling Phase

Ketika mencapai fase ini, artinya kapasitas *Cover-object* sudah cukup untuk menyembunyikan pesan yang akan disisipkan. Langkah selanjutnya adalah menyembunyikan pesan dengan cara melakukan *shuffling* pada *Global Color Table,* kemudian dilanjutkan dengan *Local Color Table* yang ada pada *frame-frame* hingga seluruh pesan telah disembunyikan.

Berikut adalah langkah-langkah pada fase ini.

1. Pisahkan pesan ke dalam blok-blok berukuran 210 byte.
2. Untuk setiap blok pesan, sisipkan pesan tersebut dengan cara melakukan Gifshuffle pada tabel warna. lakukan Gifshuffle blok pertama ke *Global Color Table*, kemudian blok kedua ke *Local Color Table* dari frame pertama, dst.
3. Setelah seluruh pesan telah disisipkan, sisipkan bilangan 0 ke *Local Color Table* frame selanjutnya yang berfungsi sebagai *terminator*.

IV. IMPLEMENTASI & PERCOBAAN

A. Hasil Implementasi

Implementasi algoritma *Extended Gifshuffle* dibuat dengan menggunakan bahasa pemrograman python. Pranala source code program disediakan di akhir makalah.

B. Hasil Percobaan & Analisis

Beberapa percobaan dilakukan untuk menguji hasil implementasi yang telah dibuat. Percobaan-percobaan tersebut dilakukan dengan menggunakan tiga buah file uji yang berbeda-beda ukurannya sebagai *embedded message*. Kemudian sebuah file GIF 'panda.gif' dengan ukuran canvas 498 * 290 yang terdiri dari 63 frame akan digunakan sebagai *cover-object*.

Gambar 5. File panda.gif yang digunakan sebagai *cover-object*

Percobaan pertama akan dilakukan dengan menggunakan file berukuran kurang dari 210 byte. Percobaan kedua akan menggunakan file berukuran lebih dari 210 byte dan kurang dari 13.230 byte (210 * 63). Kemudian Percobaan ketiga akan menggunakan file berukuran lebih dari 13.230 byte.

Percobaan pertama menggunakan file "secret1.txt" yang berukuran 27 byte. Berikut adalah hasil percobaan pertama dengan menggunakan Gifshuffle dan *Extended* GifShuffle.

```
steggif on ◿ master [?] via 🐍 v2.7.18
❯ ls -l secret1.txt
-rw-rw-r-- 1 hasim hasim 27 Des 20 16:23 secret1.txt

steggif on ◿ master [?] via 🐍 v2.7.18
❯ gifshuffle -f secret1.txt panda.gif out.gif
Message used approximately 71.62% of available space.

steggif on ◿ master [?] via 🐍 v2.7.18
❯ python3 ext-gifshuffle.py secret1.txt panda.gif out1.gif
Global Color Table :
Message used approximately 12.89% of available space.

steggif on ◿ master [?] via 🐍 v2.7.18
❯
```

Gambar 6. Percobaan pertama dengan Gifshuffle dan *Extended Gifshuffle*

Pada percobaan pertama, pesan berhasil disembunyikan baik dengan Gifshuffle maupun *Extended Gifshuffle.* Namun disini ada perbedaan pada besarnya persentase kapasitas yang digunakan. Gifshuffle melakukan *shuffling* pada tabel warna namun tanpa melakukan *padding* terlebih dahulu. File panda.gif memiliki warna unik kurang dari 256 dan hanya bisa menampung 37 byte. Sementara *Extended Gifshuffle* melakukan *padding* terlebih dahulu sehingga kapasitasnya adalah 210 byte untuk setiap tabel warna.

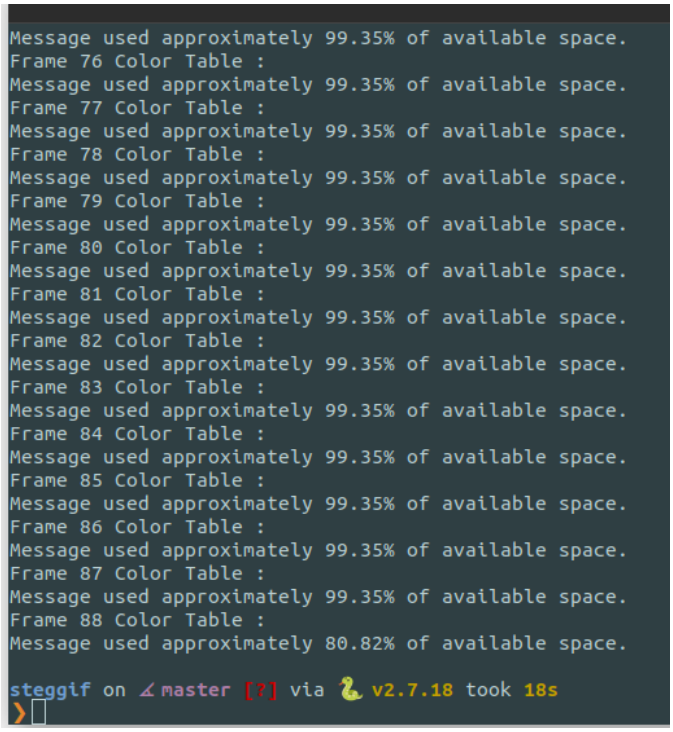
```
steggif on ◿ master [?] via 🐍 v2.7.18
❯ ls -l secret2.txt
-rw-rw-r-- 1 hasim hasim 5595 Des 20 16:50 secret2.txt

steggif on ◿ master [?] via 🐍 v2.7.18
❯ gifshuffle -f secret2.txt panda.gif out.gif
Message exceeded available space by approximately 14672.61%.

steggif on ◿ master [?] via 🐍 v2.7.18
❯ python3 ext-gifshuffle.py secret2.txt panda.gif out2.gif
Global Color Table :
Message used approximately 99.82% of available space.
Frame 1 Color Table :
Message used approximately 99.82% of available space.
Frame 2 Color Table :
Message used approximately 99.82% of available space.
Frame 3 Color Table :
Message used approximately 99.82% of available space.
Frame 4 Color Table :
Message used approximately 99.82% of available space.
Frame 5 Color Table :
Message used approximately 99.82% of available space.
Frame 6 Color Table :
Message used approximately 99.82% of available space.
Frame 7 Color Table :
Message used approximately 99.82% of available space.
Frame 8 Color Table :
Message used approximately 99.82% of available space.
Frame 9 Color Table :
Message used approximately 99.82% of available space.
Frame 10 Color Table :
Message used approximately 99.82% of available space.
Frame 11 Color Table :
Message used approximately 99.82% of available space.
Frame 12 Color Table :
Message used approximately 99.82% of available space.
Frame 13 Color Table :
Message used approximately 99.82% of available space.
Frame 14 Color Table :
Message used approximately 99.82% of available space.
Frame 15 Color Table :
Message used approximately 99.82% of available space.
Frame 16 Color Table :
Message used approximately 99.82% of available space.
Frame 17 Color Table :
Message used approximately 99.82% of available space.
Frame 18 Color Table :
Message used approximately 99.82% of available space.
Frame 19 Color Table :
Message used approximately 99.82% of available space.
Frame 20 Color Table :
Message used approximately 99.82% of available space.
Frame 21 Color Table :
Message used approximately 99.82% of available space.
Frame 22 Color Table :
Message used approximately 99.82% of available space.
Frame 23 Color Table :
Message used approximately 99.82% of available space.
Frame 24 Color Table :
Message used approximately 99.82% of available space.
Frame 25 Color Table :
Message used approximately 99.82% of available space.
Frame 26 Color Table :
Message used approximately 64.19% of available space.

steggif on ◿ master [?] via 🐍 v2.7.18 took 6s
❯
```

Gambar 7. Percobaan kedua dengan Gifshuffle dan *Extended Gifshuffle*

Percobaan kedua dilakukan dengan menggunakan file "secret2.txt" yang berukuran 5595 byte. Hasil percobaan kedua dengan menggunakan *Gifshuffle* dan *Extended GifShuffle* dapat dilihat pada Gambar 7.

Pada percobaan kedua, pesan tidak berhasil disembunyikan dengan *Gifshuffle* tetapi berhasil dengan *Extended Gifshuffle.* Pesan tidak berhasil disembunyikan dengan Gifshuffle karena kapasitas maksimum untuk Gifshuffle hanyalah 210 byte, yakni hanya dengan menggunakan *Global Color Table*. Sementara *Extended* Gifshuffle mampu menyembunyikan pesan yang berukuran lebih dari 210 byte karena tidak hanya menggunakan *Global Color Table* tetapi juga dapat menggunakan *Local Color Table* yang ada pada setiap frame. Pada kasus ini tabel warna yang digunakan adalah *Global Color Table* dan *Local Color Table* dari frame 1 hingga frame 26.

Percobaan ketiga dilakukan dengan menggunakan file "secret3.txt" yang berukuran 18650 byte. Hasil Percobaan ketiga dapat dilihat pada gambar di bawah ini.



Gambar 8. Percobaan ketiga dengan *Extended Gifshuffle.*

Pada percobaan ketiga, *Extended Gifshuffle* berhasil menyembunyikan pesan ke dalam GIF. Dapat dilihat pada Gambar 8 bahwa terjadi fase grow untuk menambah kapasitas pesan yang dapat disisipkan, ditandai dengan banyaknya frame yang dipakai sebanyak 88, melebihi banyaknya frame awal.

V. KESIMPULAN

Algoritma *Extended Gifshuffle* yang diajukan di dalam makalah ini merupakan ekstensi dari algoritma Gifshuffle. Algoritma *Extended Gifshuffle* diajukan dalam rangka mengatasi persoalan yang ada pada algoritma Gifshuffle yakni mengenai kapasitas pesan yang dapat ditampung oleh *Cover-object* terbatas pada 210 byte. Setelah dilakukan tiga kali percobaan, diperoleh hasil bahwa algoritma *Extended Gifshuffle* mampu menyembunyikan tiga buah file uji yang berbeda ke dalam file GIF dimana salah satu file uji tersebut berukuran sekitar 18KB. Dengan demikian, dapat diperoleh kesimpulan bahwa algoritma *Extended Gifshuffle* dapat digunakan sebagai alternatif untuk mengatasi persoalan *capacity* yang ada pada algoritma *Gifshuffle* sebelumnya.

PRANALA SOURCE CODE PROGRAM

https://github.com/farishasim/extended-gifshuffle

UCAPAN TERIMA KASIH

Puji syukur kepada Allah atas rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah dengan judul "Implementasi Steganografi pada File GIF dengan *Extended Gifshuffle*". Penulis juga menghaturkan ucapan terima kasih kepada berbagai pihak yang sudah membantu dalam penulisan makalah ini, kepada Dr. Ir. Rinaldi Munir, MT., selaku dosen pengampu mata kuliah IF4020 Kriptografi semester 1 tahun ajaran 2021/2022, juga kepada pihak-pihak lainnya yang tidak bisa disebutkan satu-persatu.

REFERENSI

[1] R. Munir, "Steganografi Prolog," no. Bagian 1, pp. 1–42, 2015.

[2] K. M. Miltner and T. Highfield, "Never gonna GIF you up: Analyzing the cultural significance of the animated GIF," *Soc. Media Soc.*, vol. 3, no. 3, 2017.

[3] GIFLIB, "What's In A GIF." [Online]. Available: http://giflib.sourceforge.net/whatsinagif/bits_and_bytes.html.

[4] R. Munir, "Application of the modified EzStego algorithm for hiding secret messages in the animated GIF images," *Proceeding - 2016 2nd Int. Conf. Sci. Inf. Technol. ICSITech 2016 Inf. Sci. Green Soc. Environ.*, pp. 58–62, 2017.

[5] O'Reilly, "GIF File Format Summary." [Online]. Available: https://www.fileformat.info/format/gif/egff.htm.

[6] M. Kwan, "The Gifshuffle Home Page," 2010. [Online]. Available: darkside.com.au/gifshuffle/.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2021

Faris Hasim Syauqi
13519050